Programación de aplicaciones para drones con el entorno software JdeRobot

José M. Cañas, José A. Fernández, Diego Jiménez, Jorge Vela
Universidad Rey Juan Carlos

Resumen

Un enfoque muy flexible para sacar partido de los drones es verlos como robots aéreos. Robots que tienen sensores, actuadores y computadoras. Sobre ese hardware la inteligencia del drone se materializa en su software, tanto los drivers como principalmente las aplicaciones.

JdeRobot es un entorno de software libre para robótica y visión artificial, financiado parcialmente por Google. Incluye drivers para cuadricópteros como el ArDrone2 (con o sin GPS), el 3DR Solo drone o aviones con estabilizadora tipo MAVlink, cámaras a bordo, etc.. Permite además utilizar drones en el simulador Gazebo, standard en la comunidad robótica, y afinar ahí las aplicaciones. Incluye igualmente herramientas para drones como visores, teleoperador desde un teléfono móvil e interfaz gráfico para especificar misiones como secuencias de puntos de paso sobre un mapa. En JdeRobot se pueden desarrollar fácilmente comportamientos más sofisticados como despegue y aterrizaje sobre balizas usando visión, de seguimiento de objetos, de autolocalización visual o de navegación autónoma.

El entorno software es distribuido, orientado a componentes y permite integrar una o varias computadoras, por ejemplo a bordo del drone y las de tierra. Las aplicaciones se pueden desarrollar en varios lenguajes como C++ y Python. Los componentes interoperan entre sí aunque estén escritos en lenguajes distintos.

En este trabajo se presentan las herramientas de JdeRobot para drones y varias aplicaciones de ejemplo.

1 INTRODUCCIÓN

En los últimos años hay un interés creciente en el uso de Vehículos Aéreos No-tripulados (Unmanned Aerial Vehicles, UAVs), con aplicaciones como la construcción de mapas 3D, tareas militares, seguridad, inspección [13], audiovisuales o agricultura de precisión. Quizá los cuadricópteros son los vehículos aéreos más populares actualmente, que tienen un coste relativamente barato. Hay múltiples líneas de investigación abiertas y proyectos con UAVs, incluyendo los prototipos avanzados de grandes multinacionales como el proyecto Wing de Google o el Prime Air de Amazon. Este interés creciente en la robótica aérea también se refleja en varias competiciones internacionales como *Int. Micro Air Vehicle Indoor Flight Competition* IMAV[1], *DronesForGood*[2] o *Mohamed Bin Zayed Int. Robotics Challenge*[3] (MBZIRC) que impulsan la investigación y desarrollo de tecnología para UAVs.

Para realizar muchas de las tareas anteriores los vehículos aéreos han trabajar en parte autónomamente, sin la supervisión constante de operadores humanos [1]. Varias de las capacidades deseables en este contexto son el recorrido de rutas tanto en interiores como en exteriores, la autolocalización, el aterrizaje preciso, la detección y evitación de obstáculos entre otras. Un ejemplo importante de navegación autónoma es el recorrido de rutas en 3D con drones, que se aborda en numerosos trabajos [5,14]. Para esa navegación la autolocalización es una capacidad clave. En exteriores el sensor fundamental de localización es el GPS, mientras que en interiores la solución puede venir de sistemas de captura de movimiento, instalados en el escenario [8, 11] o de visión

1 http://www.imavs.org
2 http://www.dronesforgood.ae/
3 http://www.mbzirc.com

artificial a bordo [3, 18, 21], incluyendo opcionalmente IMUs. Esa autolocalización se complementa con un sistema de control que gobierna los movimientos del drone. Existen varias aproximaciones para conseguir ese pilotaje: controladores PID, controladores borrosos, control basado en modelo interno [6], control basado en modelo predictivo (MPC) [7], control basado en visión, etc. El control visual se utiliza por ejemplo en el aterrizaje de precisión [14].

Los UAVs se pueden ver como robots aéreos, sistemas robóticos dotados de hardware y software. En el hardware incorporan sensores, actuadores, ordenadores y elementos de comunicación. En el software, al igual que con otros robots, las aplicaciones se suelen programar usando algún *middleware* o entorno que simplifica el desarrollo. Existen varios entornos software para UAVs que ayudan al desarrollo de esas capacidades y proporcionan soporte para diferentes plataformas voladoras. Por ejemplo AEROSTACK [19, 20], PX4 Flight Stack[4] , APM Flight Stack[5], Telekyb [4], entorno del cuadricóptero Hector [8], más allá de otros entornos robóticos generalistas como ROS.

En la primera sección de este artículo se describe el entorno software JdeRobot de robótica y visión de manera genérica. En las siguientes de presentan las piezas software que ofrece para la programación de aplicaciones con drones: drivers y herramientas. Para ilustrar la capacidad del entorno se introducen tres aplicaciones desarrolladas con él, a modo de ejemplo. Se recapitulan las conclusiones en la sección final.

2 Entorno software JdeRobot

.JdeRobot es un *middleware* para el desarrollo de aplicaciones con robots y visión artificial. Esta plataforma fue creada por el Laboratorio de Robótica de la Universidad Rey Juan Carlos en 2003, es software libre, está licenciado como GPLv3 3 y disponible en abierto en el repositorio de GitHub[6].

En este entorno las aplicaciones están compuestas por uno o varios componentes, que se ejecutan como procesos y que interoperan entre sí a través del middleware de comunicaciones, tanto ICE (Internet Communications Engine, de ZeroC) como ROS-messages (de la Open Source Robots Foundation). Los componentes pueden estar escritos en lenguajes diferentes como C++, Python o incluso JavaScript, que interoperan sin problema.

Los componentes que interactúan directamente con los sensores y actuadores del robot se llaman *drivers*, recogen los datos sensoriales o envía órdenes a los actuadores. El entorno ofrece actualmente drivers que dan soporte a gran variedad de dispositivos como el robot Pioneer de MobileRobotics Inc., el robot TurtleBot de Yujin Robot, el humanoide Pepper de SoftBank, webcams, las cámaras RGBD como Kinect o Xtion, los escáneres laser LMS de SICK y URG de Hokuyo, los simuladores Stage y Gazebo, etc.

En las aplicaciones en JdeRobot se utilizan bibliotecas de software libre que son estandares de facto en su campo como OpenCV para visión, PCL para manejo de nubes de puntos o Eigen para álgebra. Además, JdeRobot es compatible con ROS, en particular con la versión ROS-Kinetic, por lo que las aplicaciones puede incorporar fluidamente nodos de ROS y conectarse a ellos.

Los componentes de la aplicación robótica suelen llevar en su código las funciones perceptivas, procesamiento de señales o la lógica de control e inteligencia del robot. Típicamente llevan a cabo

4 http://px4.io
5 http://ardupilot.com
6 https://github.com/JdeRobot/JdeRobot

diferentes tareas en tiempo real. El acceso a datos sensoriales o comandos a los actuadores se realiza, desde la perspectiva del programador, invocando un sencillo método local que por debajo materializa la comunicación con el componente driver respectivo.

JdeRobot también incluye también varias herramientas y varias bibliotecas específicas que encapsulan funcionalidad reutilizable. Por ejemplo, proporciona el entorno educativo JdeRobot-Academy[7], que es un conjunto de quince prácticas docentes diseñadas para enseñar robótica en la universidad, en las carreras de ingeniería principalmente. Tiene varias prácticas de visión artificial (control visual, filtros, reconstrucción 3D...), de coches autónomos (navegación local, navegación global…) y de programación drones. En ellas se emplea el simulador Gazebo y el lenguaje Python. Cada práctica ha sido diseñada como un nodo académico que aloja el código del estudiante y que resuelve funcionalidad auxiliar como la interfaz gráfica o código de apoyo. Con este diseño el alumno se puede concentrar en el algoritmo respectivo, dando por resueltos otros aspectos que son necesarios para que la práctica funcione pero que no son interesantes desde el punto de vista docente.

3 Drivers para drones

Uno de los aportes del entorno JdeRobot respecto de los drones es un conjunto de cuatro drivers que proporcionan acceso a los sensores y actuadores de varias plataformas robóticas aéreas. Los drivers aceptan un conjunto de mensajes a través de los cuales las aplicaciones pueden recoger en tiempo real las medidas de los sensores embarcados en el robot aéreo y pueden ordenarle movimientos.

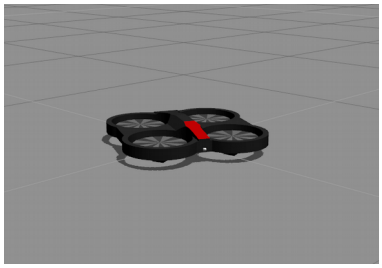
Los sensores típicos incluyen unidades inerciales (Inertial Measurement Unit, IMU), altímetros, GPS en algunos modelos, y una o dos cámaras. Para los dos primeros JdeRobot define una estructura de datos que almacena posición-orientación en tres dimensiones. Incluye tres coordenadas cartesianas y un cuaternión para reflejar la orientación. Los drivers recogen información de orientación de los GPS, IMU, giróscopos, etc. y la convierte a cuaterniones que tiene ciertas ventajas computaciones frente a otras representaciones como Cabeceo-Alabeo-Guiñada (Yaw-Pitch-Roll). Igualmente, la latitud-longitud-altura del GPS son traducidos a X-Y-Z respecto de cierto sistema referencia absoluto.

Para las cámaras, JdeRobot tiene una interfaz que proporciona las imágenes RGB en formato estandard. Esto facilita la utilización de visores ya existentes para otros robots, por ejemplo sintonizadores de filtros de color.

En cuanto a los comandos de movimiento, típicamente un drone tiene de tres niveles: (a) órdenes directas a cada motor de hélice independientemente, (b) comandos de velocidad para el drone entero como un sólido rígido (velocidad de giro, de ascenso, de avance…) y (b) comandos de posición global, secuencias de puntos de paso, también conocidos como misiones. Además, se incorporan comandos especiales como las órdenes de despegue, de aterrizaje o cambios de modo. El nivel (a) habitualmente no está disponible para los programadores, y lo utiliza el estabilizador incluido por el fabricante. El nivel (b) es muy útil para tener un control fino del UAV, incluso en interiores, aunque no todos los grados de libertad de movimiento en 3D son controlables independientemente. Es habitual que sí se incluyan los básicos como velocidad de avance frontal/retroceso, velocidad lateral, velocidad de ascenso/descenso y velocidad de giro en horizontal. El nivel (c) se aplica sólo en exteriores y es el que suelen ofrecer aplicaciones de usuario sencillas.

7 http://jderobot.org/JdeRobot-Academy

Primero, el driver *ArDroneServer* da soporte al ArDrone2.0 de Parrot, que se ha usado principalmente como juguete y en interiores. Este cuadricóptero tiene dos cámaras, una frontal y otra cenital. El driver permite a las aplicaciones cambiar explícitamente entre una u otra.



Segundo, el driver *MAVProxy* da soporte al Solo Drone de 3DRobotics, que es una plataforma más sólida, con motores más potentes capaces de desplazamientos fiables en exteriores. Este driver emplea MAVlink (Micro Air Vehicle Link), que es un protocolo de comunicacones con pequeños vehículos no tripulados. Se emplea principalmente para la comunicación entre ellos y las estaciones de control en tierra. También para la transmisión de la orientación del UAV, su posición GPS y velocidad. Hay en el mercado varios drones que incorporan placas estabilizadoras que soportan MAVlink, que por tanto aceptan comandos estandarizados de movimiento, lecturas de GPS, etc..

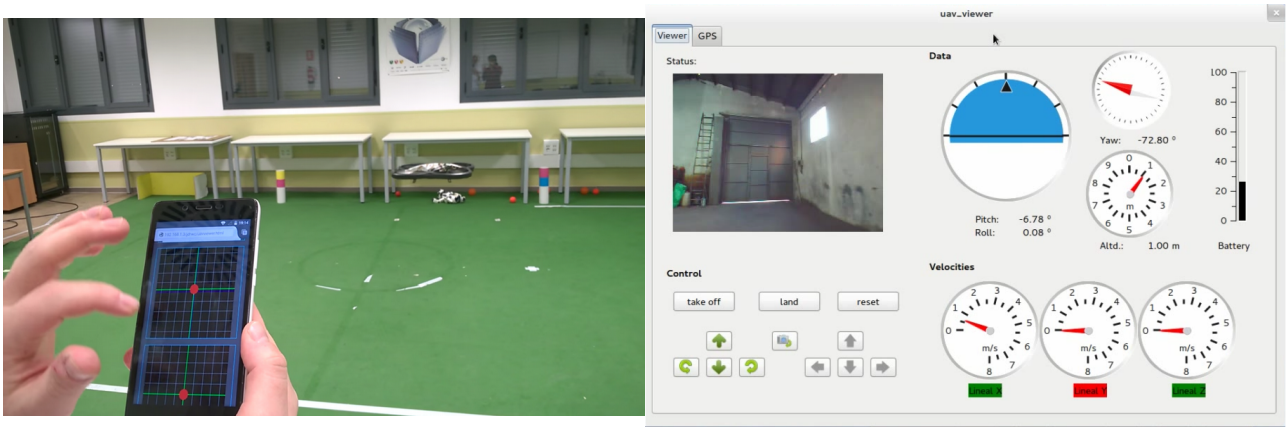
Tercero, el driver *MAVLinkServer* utiliza igualmente el protocolo MAVlink, pero hace énfasis en el nivel (c) de actuadores, las secuencias de puntos de paso. Ha sido probado con un UAV avión Bix3.

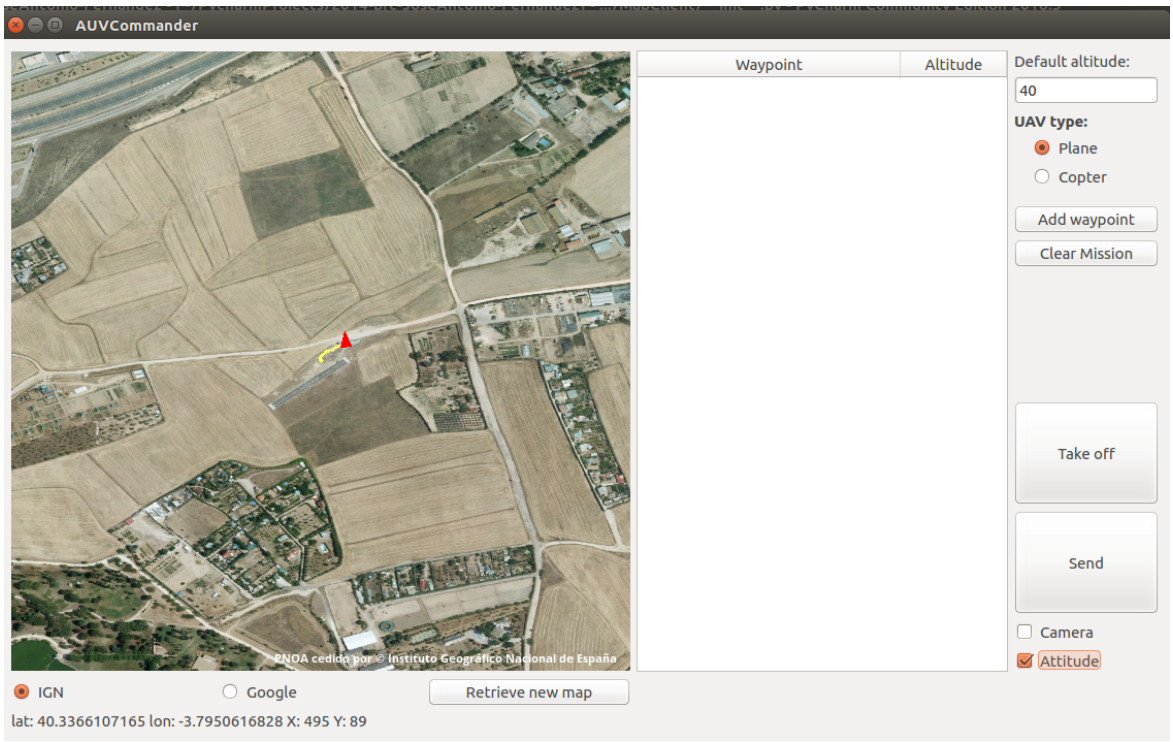
En cuarto lugar, JdeRobot incorpora driver para un cuadricóptero simulado. Este driver tiene la forma de plugin en C++ para el simulador Gazebo, permite programar aplicaciones probándolas y madurándolas primero en simulador, y sólo cuando están razonablemente fiables, llevarlas al robot real.

El diseño de estos drivers tiene la capacidad de funcionar bien en un ordenador conectado por radio o wifi al drone, o bien completamente a bordo, embarcado en algún microprocesador (por ejemplo Intel Computer Stick o RaspBerry-PI) al que se conecta la placa estabilizadora y la cámara.

4 Herramientas

Además de los drivers concretos, el entorno JdeRobot ofrece varias herramientas que aportan funcionalidad más allá del acceso básico a sensores y actuadores. Ayudan a acelerar el desarrollo de aplicaciones para robots aéreos.

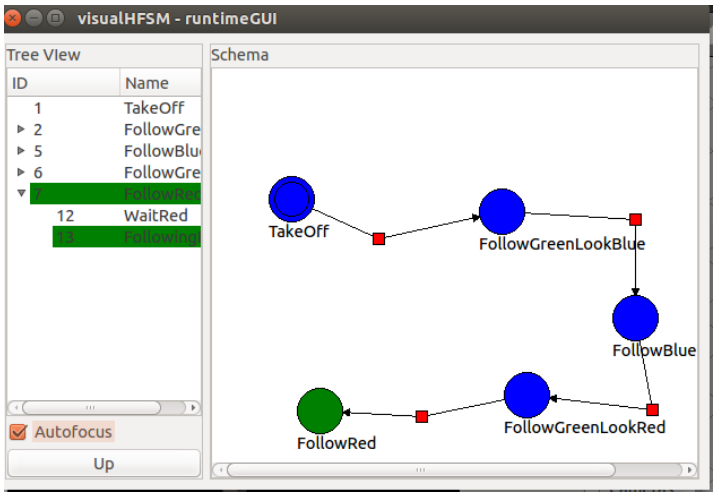




Primero, el componente *UAVviewer* [12], programado en Python, permite al usuario ver las imágenes de la cámara a bordo, un horizonte virtual, las velocidades, la altimetría, hacer que despegue o aterrice, teleoperarlo o cambiar de cámara activa desde el ordenador. Se incluye también una versión web de esta herrramienta *UAVviewer-web*, programada en JavaScript, que permite teleoperarlo desde un navegador web, incluidos los disponibles en los teléfonos móviles.

Segundo, el componente *UAVCommander* [2] permite visualizar un mapa georeferenciado en la pantalla y especificar y modificar misiones de navegación al drone picando en esos mapas para marcar la secuencia de puntos de paso. Además, gracias al GPS, muestra la posición instantánea del drone dentro de ese mapa.

Una capacidad clave para que un drone pueda moverse autónomamente en interiores es la autolocalización, que sepa estimar su posición tridimensional en tiempo real. Aquí JdeRobot ofrece dos herramientas muy útiles. Por un lado el componente *Slam-Markers* [10] que estima la posición 3D a partir de un mapa conocido. El mapa es una colección de balizas visuales (marcadores AprilTags) cuya posición 3D se conoce de antemano. Para distancias inferiores a 4 metros este componente realiza una estimación 3D fiable. Estos marcadores se pueden imprimir con facilidad y pegar en el entorno. Por otro lado, el componente *Slam-SDVL* estima la posición 3D sin necesidad de tener un mapa de antemano ni balizas, sólo necesita que el escenario tenga textura. Este componente utiliza algoritmo sofisticado de autolocalización visual [16], está optimizado para ejecutar en tiempo real incluso en procesadores limitados. Estos dos componentes han sido desarrollados para ser ejecutados tanto en robots con ruedas como en humanoides y en drones. Son un ejemplo de los beneficios de contemplar a los drones como robots aéreos.



Otra herramienta poderosa disponible es *VisualStates* [17], que permite especificar la lógica del comportamiento del drone en términos de estados y transiciones, de autómata finito de estados. Este paradigma tiene mayor expresividad que los sistemas reactivos puros, permite expresar comportamientos más variados y escalar en complejidad de los comportamientos generables. De nuevo, esta herramienta nació en el contexto de los robots con ruedas y de los humanoides de la RoboCup, pero sirve sin fisuras para los robots aéreos, sin más que configurar en el autómata el acceso a sus sensores y actuadores.

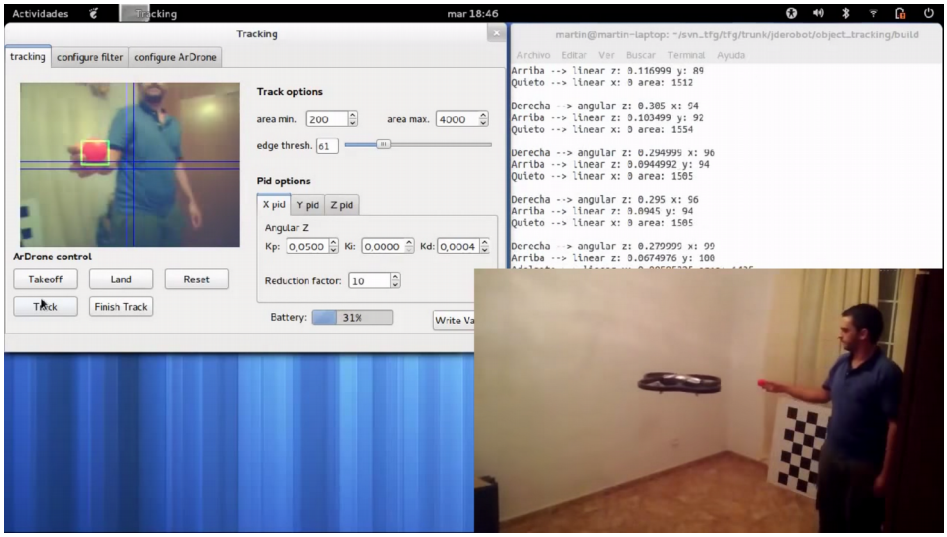
Otro instrumento genérico de JdeRobot que es de utilidad en el desarrollo con drones es el calibrador de cámaras, basado en el calibrador de la biblioteca OpenCV.

5 Aplicaciones de ejemplo

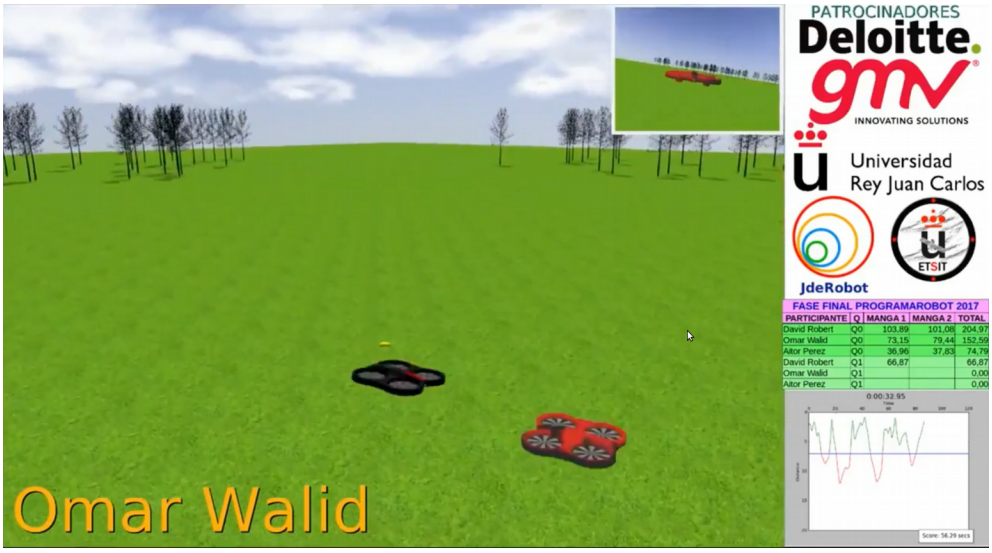
Una vez descritos los drivers y herramientas que ofrece JdeRobot para facilitar la programación de aplicaciones para drones, en esta sección se describen a modo ilustrativo tres aplicaciones de comportamiento autónomo desarrolladas con esa infraestructura.

Se han desarrollado varias aplicaciones de control visual, de seguimiento de objetos. En [12] se describen dos de ellas en las que el robot real ArDrone2.0 persigue a una pelota rosa en tres dimensiones utilizando la cámara frontal o a un objeto móvil en el suelo empleando la cámara

ventral. Ambas se han resuelto con comportamientos reactivos iterativos. La percepción se ciñe a identificar en la imagen el estímulo de interés, básicamente empleando filtro de color. La actuación se materializa en ambos con un controlador PID que busca centrar el estímulo en la imagen, y se incluyen casos de búsqueda para cuando el estímulo no aparezca en la imagen.



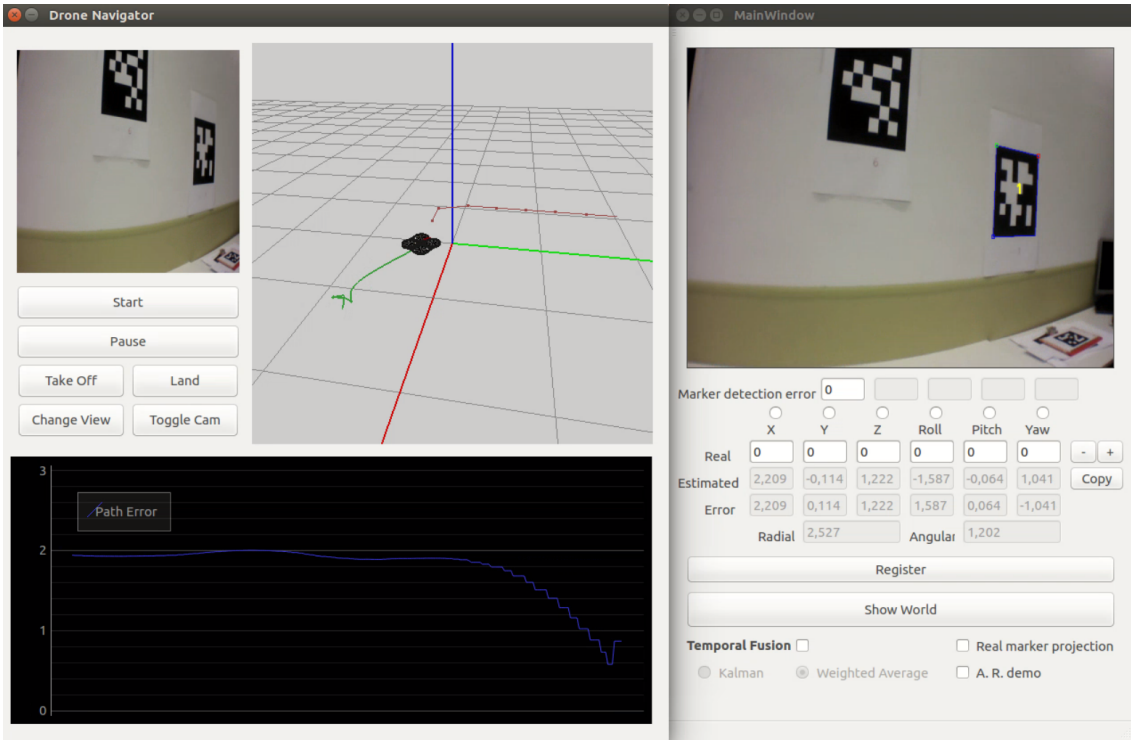
También se ha desarrollado otro comportamiento de seguimiento para el campeonato de programación de robots Program-A-Robot[8]. En la prueba de la edición de 2017 los participantes tenían que programar un drone-gato para que buscara y persiguiera a un drone-ratón que ponía la organización.



Otra aplicación ilustrativa realizada consigue que un drone recorra una ruta en 3D planificada de antemano [22] en entornos de interiores. En este comportamiento se ha utilizado el nodo *Slam-markers* para proporcionar una estimación de posición. Además, incluye un pilotaje que corrige

8 http://jderobot.org/Campeonato-programacion-de-robots

desviaciones respecto de la ruta planificada, tanto girando el drone en horizontal como haciendo que se eleve o descienda.



La tercera aplicación ilustrativa consigue un aterrizaje preciso visual justo encima de una baliza en el suelo[9]. Es otro ejemplo de control visual. La baliza empleada tiene forma de cuadrícula arlequinada con cuatro cuadrados de dos colores diferentes. La percepción utiliza OpenCV para identificar los colores relevantes, las fronteras entre ellos y la cruceta. El control es uno basado en casos que incluye búsqueda si se pierde, aproximación si la baliza se ve sólo parcialmente y el centrado.

6 Conclusiones

En este artículo se han presentado las ventajas de utilizar un entorno de software robótico como JdeRobot en el desarrollo de aplicaciones para drones, que se miran como robots aéreos. La primera es utilizar drivers existentes que ofrecen a las aplicaciones acceso a sus sensores y actuadores, típicamente cámaras, GPS, IMU y motores. La segunda es aprovecharse de herramientas ya disponibles en el entorno que simplifican su manejo y la programación de aplicaciones. Por un lado, instrumentos específicos como visores, teleoperadores web y consignar de misiones de navegación como secuencias de puntos de paso. Por otro lado, instrumentos genéricos de robótica que son igualmente aplicables a los drones. Por ejemplo los componentes de autolocalización visual o la herramienta de programación de robots con autómatas de estado finito.

Además, se han introducido varias aplicaciones a modo de ejemplo que utilizan esa plataforma. Tres de seguimiento visual de objetos, una de recorrido de una ruta tridimensional en interiores y una de aterrizaje preciso autónomo encima de una baliza visual en el suelo.

9 http://jderobot.org/Jvela-tfg

Como líneas futuras, en JdeRobot se está trabajando en incorporar una herramienta que permite la programación visual con lenguaje Scratch de robots con ruedas como los TurtleBot. Se están incluyendo bloques visuales que permiten programar drones ordenándoles movimiento tridimensional, recoger los datos del GPS y el procesamiento básico de las imágenes. También se está trabajando en una nueva versión de la aplicación de seguimiento de rutas, pero ahora sin requerir balizas en el escenario, utilizando el componente de autocalización *slam-SDVL*.

7 Referencias

[1] Apvrille, L., Dugelay, J.L., Ranft, B.: Indoor autonomous navigation of low-cost mavs using landmarks and 3d perception. Proc. Ocean and Coastal Observation, Sensors and Observing Systems. 2013

[2] J.A. Fernández, Navegación por posición para un avión autónomo con JdeRobot, Proyecto Fin de Carrera, U. Rey Juan Carlos, 2017.

[3] Forster, C., Pizzoli, M., Scaramuzza, D.: SVO: Fast semi-direct monocular visual odometry, in Robotics and Automation (ICRA), 2014 IEEE International Conference on. IEEE, 2014, pp. 15–22.

[4] V. Grabe, M. Riedel, H.H. Bulthoff, P.R. Giordano, and A. Franchi: The telekyb framework for a modular and extendible ros-based quadrotor control. In Mobile Robots (ECMR), 2013 European Conference on, pages 19–25, Sept 2013

[5] Hehn, M., D'Andrea, R.: Real-Time Trajectory Generation for Quadrocopters, IEEE TRANSACTIONS ON ROBOTICS, 31(4), 2015

[6] A. Hernández, C. Copot and R. De Keyser Tudor Vlas and Ioan Nascu Identification and Path Following Control of an AR.Drone Quadrotor 2013

[7] A. Hernandez, H. Murcia, C. Copot, R. De Keyser: Model Predictive PathFollowing Control of an AR.Drone Quadrotor Memorias del XVI Congreso Latinoamericano de Control Automático, CLCA 2014, Octubre 14-17, 2014. Cancún, Quintana Roo, México

[8] Jiménez, J., Zell, A.: Framework for Autonomous On-board Navigation with the AR.Drone, Journal of Intelligent & Robotic Systems, January 2014, Volume 73, Issue 1–4, pp 401–412

[9] S. Kohlbrecher, J. Meyer, T. Graber, K Petersen, O. von Stryk, and U Klingauf. RobocupRescue 2014-robot league team hector darmstadt (Germany). RoboCupRescue 2014, 2014.

[10] López-Cerón, A., Cañas, J.M.: Accuracy analysis of marker-based 3D visual localization XXXVII Jornadas de Automática, Madrid, 2016

[11] Lupashin, S., Hehn, M., Mueller, M., Schoellig, A., Sherback, M., D'Andrea, R.: A platform for aerial robotics research and demonstration: The Flying Machine Arena, Mechatronics 24 (2014) 41–54

[12] A. Martín, Navegación visual en un cuadricóptero para el seguimiento de objetos, Trabajo Fin de Grado, U.Rey Juan Carlos, 2014.

[13] Nikolic, J., Leutenegger, S., Burri, M., Huerzeler, C., Rehder, J., Siegwart, R.: A. UAV System for Inspection of Industrial Facilities, IEEE Aerospace Conference, 2013 10.1109/AERO.2013.6496959

[14] T. Nguyen, G. K. I. Mann and R. G. Gosine Vision-Based Qualitative PathFollowing Control of Quadrotor Aerial Vehicle 2014 International Conference on Unmanned Aircraft Systems (ICUAS) May 27-30, 2014. Orlando, FL, USA

[15] Pestana, J., Sanchez-Lopez, J.L., de la Puente, P., Carrio, A., Campoy, P.: A vision-based quadrotor multi-robot solution for the indoor autonomy challenge of the 2013 international micro air vehicle competition: Journal of Intelligent & Robotic Systems, pages 1–20, 2015.

[16] E. Perdices, Técnicas para la localización visual robusta de robots en tiempo real con y sin mapas, Tesis doctoral, U. Rey Juan Carlos, 2017

[17] S. Rey, VisualHFSM 5.0, Trabajo Fin de Grado, U. Rey Juan Carlos, 2016.

[18] A. Rodriguez-Ramos, C. Sampedro, A. Carrio, H. Bavle, R. A. Suarez Fernandez, Z. Milosevic, P. Campoy: A Monocular Pose Estimation Strategy for UAV Autonomous Navigation in GNSS-denied Environments IMAV 2016

[19] J. L. Sanchez-Lopez, J. Pestana, P. Puente, and P. Campoy: A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation. Journal of Intelligent & Robotic Systems, pages 1–19, 2015.

[20] J.L. Sanchez-Lopez, R. A. Suárez Fernández, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy: AEROSTACK: An Architecture and Open-Source Software Framework for Aerial Robotics 2016 International Conference on Unmanned Aircraft Systems (ICUAS) June 7-10, 2016. Arlington, VA USA

[21] Wu, A., Johnson, E.N., Kaess, M., Dellaert, F., Chowdhary, G.: Autonomous Flight in GPS-Denied Environments Using Monocular Vision and Inertial Sensors. J. Aerospace Inf. Sys.. 2013 Apr 1;10(4):172-86.

[22] M. Zafra, J.M. Cañas, Fine 3D path following of a quadcopter, Actas de ROBOT2017, 2017.